LinuxFocus article number 225
http://linuxfocus.org

"Chrooter" tous les Services sous Linux

par Mark Nielsen (homepage)

L´auteur:
Mark travaille comme consultant indépendant et donne de son temps à des causes telles que GNUJobs.com, l'écriture d'articles, la création de logiciel libre, et le bénévolat pour eastmont.net.

Traduit en Français par:
Georges Tarbouriech
<georges.t/at/linuxfocus.org>

Résumé:

NDT : Ne cherchez pas le verbe "chrooter" (composé à partir de la commande Unix "chroot") dans le Petit Robert ou le Petit Larousse : il n'y est pas ! Pourtant, vous le trouverez conjugué tout au long de cet article. C'est un choix délibéré pour faciliter la compréhension mais je reconnais que c'est très laid. Je réclame donc l'indulgence du lecteur pour cet horrible anglicisme. Pour me faire pardonner, il sera toujours mis entre guillemets.

Les services "chrootés" améliorent la sécurité en limitant les dommages que pourrait causer un intrus ayant pénétré votre système.

Introduction

Qu'est-ce que chroot ? Chroot redéfinit l'univers de fonctionnement d'un programme. Plus précisément, il détermine un nouveau répertoire "ROOT" ou "/" pour un programme ou une session. Schématiquement, tout ce qui est à l'extérieur du répertoire "chrooté" n'existe pas pour un programme ou un shell.

Pourquoi est-ce utile ? Si quelqu'un s'introduit dans votre ordinateur, il ne pourra pas voir la totalité des fichiers de votre système. Le fait de ne pas voir vos fichiers limite les commandes qu'il peut lancer et ne lui donne pas la possibilité d'exploiter des fichiers qui seraient vulnérables. L'inconvénient majeur, c'est que ça n'empêche pas l'analyse des connexions réseau ou autre. Ainsi, vous devrez faire bien d'autres choses qui n'entrent pas vraiment dans le cadre de cet article :

- Sécuriser vos ports réseau.
- Exécuter tous les services sous un compte qui ne soit pas root. Tous vos services doivent en plus être "chrootés".
- Transférer les logs système sur une autre machine.

- Analyser les fichiers de logs.
- Rechercher les tentatives de détection de ports ouverts sur votre machine.
- Limiter les ressources cpu et mémoire pour un service.
- Activer les quotas pour les comptes utilisateurs.

La raison pour laquelle je considère chroot (pour un service non root) comme une ligne de défense vient du fait que si quelqu'un s'introduit dans votre machine sous un compte non root et si aucun fichier n'est disponible lui permettant de passer root, les dommages pouvant être causés, seront limités à la zone dans laquelle il a pu pénétrer. De même, si le compte root est essentiellement propriétaire de la zone dans laquelle il s'introduit, les options d'attaque sont moindres. Il est évident que vous avez un sérieux problème si un intrus peut entrer dans votre machine, mais ce n'est pas si mal de pouvoir limiter les dégâts qu'il est susceptible de causer.

RAPPELEZ-VOUS que ma manière de procéder n'est certainement pas précise à 100%. Il s'agit de ma première tentative et si tout semble bien fonctionner séparément, il devrait être facile d'affiner l'ensemble. Ce n'est que l'ébauche d'un HOWTO que je souhaite créer sur chroot.

Comment allons-nous tout "chrooter" ?

Eh bien, nous créons un répertoire, "/chroot" et nous y mettons tous les services de la manière suivante :

- Syslogd sera "chrooté" pour chaque service.
- Apache sera dans /chroot/httpd.
- Ssh sera dans /chroot/sshd.
- PostgreSQL sera dans /chroot/postmaster.
- Sendmail sera "chrooté", mais malheureusement il ne fonctionnera pas sous un compte non root.
- ntpd sera dans /chroot/ntpd
- named sera dans /chroot/named

Chaque service devrait se retrouver totalement isolé.

Mon script Perl pour créer des environnements "chrootés".

Config_Chroot.pl.txt doit être renommé Config_Chroot.pl après téléchargement. Ce script perl vous permet de lister les services en cours d'installation, de visualiser les fichiers de configuration, de configurer un service, et de démarrer ou d'arrêter les services. Voici la marche à suivre.

1. Créer le répertoire chroot.
   mkdir -p /chroot/Config/Backup
2. Télécharger Config_Chroot.pl.txt dans /chroot/Config_Chroot.pl
3. Changer la variable $Home du script perl si vous n'utilisez pas /chroot comme répertoire home.
4. Télécharger mes fichiers de configuration.

Une remarque importante : **Je n'ai testé que sur RedHat 7.2 et RedHat 6.2**.

Modifiez le script perl en fonction de votre distribution.

Je me suis retrouvé en train d'écrire un article gigantesque sur chroot, mais grâce au script Perl il est

devenu beaucoup plus court. En gros, après avoir "chrooté" de nombreux services, j'ai remarqué qu'ils avaient des fichiers et des configurations semblables devant également être "chrootés". Le meilleur moyen de savoir quels sont les fichiers à copier pour un service donné consiste à lire la page de manuel et de taper "ldd /usr/bin/file" pour les programmes utilisant des bibliothèques. Vous pouvez aussi "chrooter" le service que vous installez et le démarrer manuellement pour visualiser les erreurs éventuelles ou analyser ses fichiers de log.

En général, pour installer un service tapez ceci :

```
cd /chroot
./Config_Chroot.pl config  SERVICE
./Config_Chroot.pl install SERVICE
./Config_Chroot.pl start   SERVICE
```

"Chrooter" Ntpd

Ntpd est un simple service de distribution d'heure qui permet de synchroniser tous vos ordinateurs. Il a été très facile à "chrooter".

```
cd /chroot
# Supprimez le commentaire de la ligne suivante si vous n'utilisez pas mon
fichier de configuration.
#./Config_Chroot.pl config  ntpd
./Config_Chroot.pl install ntpd
./Config_Chroot.pl start   ntpd
```

"Chrooter" DNS ou named

C'est déjà fait, vérifiez sur
http://www.linuxdoc.org/HOWTO/Chroot-BIND8-HOWTO.html
ou
http://www.linuxdoc.org/HOWTO/Chroot-BIND-HOWTO.html

Ou, si vous voulez utiliser mon script,

```
cd /chroot
 # Supprimez le commentaire de la ligne suivante si vous n'utilisez pas mon
fichier de configuration.
#./Config_Chroot.pl config  named
./Config_Chroot.pl install named
./Config_Chroot.pl start   named
```

"Chrooter" Syslog et les services... et mes doléances.

Je veux "chrooter" syslogd. Mon problème c'est que syslogd utilise /dev/log par défaut, ce qui n'est pas vu par des services "chrootés". Par conséquent, il n'est pas facile d'obtenir des logs. Voici quelques

solutions :

- "Chrootez" syslogd pour chaque service. J'ai testé, et j'ai bien obtenu des logs. Ca ne me plaît pas puisque j'ai un service fonctionnant sous root.
- Voyons si nous pouvons nous connecter sur un site externe capable de nous fournir des logs.
- Envoyez les logs vers un fichier au lieu d'utiliser le démon syslogd. C'est probablement la meilleure option du point de vue de la sécurité, toutefois, si un intrus pénètre votre système, rien ne l'empêche de "jouer" avec les logs.
- Configurez le syslogd principal pour qu'il cherche à différents endroits l'ensemble des services. Pour cela, appliquez l'option -a à syslogd.

Mon unique solution a été de vérifier que syslogd était bien "chrooté" pour chaque service. J'aimerais trouver un moyen d'écrire des logs dans un compte non root qui aurait son propre environnement "chrooté", comme peut-être un port réseau. C'est sans doute réalisable, mais je vais m'arrêter où j'en suis et essayer de trouver une meilleure solution ultérieurement.

Si vous ne voulez pas d'un démon syslogd pour chaque service, alors ajoutez les commandes suivantes au démon principal lors de son lancement :

```
syslogd -a /chroot/SERVICE/dev/log
```

Si ssh et dns étaient actifs, ça devrait ressembler à ceci :

```
syslogd -a /chroot/ssh/dev/log -a /chroot/named/dev/log -a /dev/log
```

Dernière chose concernant syslogd, j'aimerais pouvoir l'exécuter sous un compte non root. J'ai essayé quelques trucs simples, mais ça n'a pas fonctionné et j'ai laissé tomber. Si je pouvais exécuter syslogd sous un compte non root pour chaque service, mes besoins en sécurité seraient comblés. Une solution de log externe est peut-être envisageable.

"Chrooter" Apache

Ce fut très facile à faire. Une fois réalisé, j'ai pu exécuter des scripts Perl. Maintenant, mon fichier de configuration est plutôt long puisque j'ai dû inclure les bibliothèques Perl et PostgreSQL dans la zone "chrootée". Une chose à noter, si vous vous connectez à une base de données, vérifiez que le service est fourni sur l'interface loopback 127.0.0.1 et que cet hôte est bien spécifié dans vos scripts Perl pour le module DBI. Voici comment contacter une base de données en utilisant les connexions permanentes d'Apache :

```
$dbh ||= DBI->connect('dbi:Pg:dbname=DATABASE',"","", {PrintError=>0});

if ($dbh ) {$dbh->{PrintError} = 1;}
else
  {$dbh ||= DBI->connect('dbi:Pg:dbname=DATABASE;host=127.0.0.1',"","",
      {PrintError=>1});}
```

Source: http://httpd.apache.org/dist/httpd/

Compilez et installez apache sur votre système dans /usr/local/apache. Utilisez ensuite le script perl.

```
cd /chroot
```

```
# Supprimez le commentaire de la ligne suivante si vous n'utilisez pas mon
fichier de configuration.
 # ./Config_Chroot.pl config  httpd
./Config_Chroot.pl install httpd
./Config_Chroot.pl start   httpd
```

J'ai modifié mon fichier de configuration httpd.conf de la façon suivante :

```
ExtendedStatus On

<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
</Location>

<Location /server-info>
    SetHandler server-info
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
</Location>
```

Ensuite, avec votre navigateur allez sur
http://127.0.0.1/server-status
ou
http://127.0.0.1/server-info
et vérifiez le résultat !

"Chrooter" Ssh

Tout d'abord, l'idéal serait de rediriger le port 22 de ssh sur le port 2222. Ensuite, au lancement de ssh, faites-lui écouter le port 2222 sous un compte non root. Pour la connexion ssh initiale, nous voulons des comptes sécurisés par mots de passe pour laisser aux utilisateurs la possibilité de se connecter et rien d'autre. Après leur authentification, utilisez une seconde instance de ssh sur le port 127.0.0.1:2322 qui leur permette de se connecter réellement au système -- le second ssh NE doit être à l'écoute que de l'interface loopback. Voici maintenant ce que vous devriez faire, mais nous ne le ferons pas. La seule chose que nous allons effectuer consiste à "chrooter" ssh pour cet exemple. Les exercices laissés au lecteur concernent le lancement d'un démon sshd sous un compte non root et l'installation d'un second démon sshd à l'écoute de l'interface loopback permettant aux utilisateurs d'accéder au système.

Encore une fois, nous n'allons que "chrooter" ssh et vous laisser réfléchir aux conséquences que ça implique (vous ne verrez plus la totalité du système si vous vous contentez de cette étape). Idéalement, ce serait très bien de prévoir d'enregistrer les logs de manière externe. Enfin, nous devrions utiliser OpenSSH alors que je me sers du SSH commercial pour des raisons de simplicité (ce qui n'est pas une bonne excuse).

Source: http://www.ssh.com/products/ssh/download.cfm

Installez ssh dans /usr/local/ssh_chroot. Utilisez ensuite le script Perl.

```
cd /chroot
# Supprimez le commentaire de la ligne suivante si vous n’utilisez pas mon
fichier de configuration.
 # ./Config_Chroot.pl config  sshd
./Config_Chroot.pl install sshd
./Config_Chroot.pl start   sshd
```

Je pense que l’une des très bonnes raisons de mettre ssh dans un environnement "chrooté" vient du fait que si vous souhaitez remplacer un serveur ftp, les visiteurs auront un accès limité à cette zone. Rsync et SCP fonctionnent très bien ensemble afin de permettre aux utilisateurs de charger des fichiers. Je n’aime pas trop l’idée d’un serveur ftp permettant d’entrer dans le système. De nombreux serveurs ftp sont "chrootés" mais ils transmettent encore les mots de passe en clair, ce que je n’aime pas vraiment.

"Chrooter" PostgreSQL

Ce fut aussi simple que pour Perl, sauf que davantage de bibliothèques ont été requises. Dans l’ensemble, ce n’était vraiment pas difficile à faire. J’ai dû ouvrir PostgreSQL au réseau, mais seulement sur l’interface loopback. Comme il était "chrooté", les autres services "chrootés" n’y avaient pas accès, comme par exemple le serveur apache. J’ai compilé Perl dans PostgreSQL et j’ai dû ainsi ajouter beaucoup de choses concernant Perl dans mon fichier de configuration.

Source: ftp://ftp.us.postgresql.org/source/v7.1.3/postgresql-7.1.3.tar.gz

Compilez et installez postgres sur votre système dans /usr/local/postgres. Utilisez ensuite le script Perl.

```
cd /chroot
 # Supprimez le commentaire de la ligne suivante si vous n’utilisez pas mon
fichier de configuration.
 # ./Config_Chroot.pl config  postgres
./Config_Chroot.pl install postgres
./Config_Chroot.pl start   postgres
```

"Chrooter" Sendmail

Exécutez mon script.

```
cd /chroot
# Supprimez le commentaire de la ligne suivante si vous n’utilisez pas mon
fichier de configuration.
 # ./Config_Chroot.pl config  sendmail
./Config_Chroot.pl install sendmail
./Config_Chroot.pl start   sendmail
```

Maintenant, est-ce qu’il reste des inconvénients ? Oui. Il fonctionne toujours en tant que root. Malédiction ! De plus, certains fichiers sont recréés par /etc/rc.d/init.d/sendmail lorsqu’il est démarré. Mon script ne gère pas cela. Chaque fois que vous modifiez sendmail dans /etc/mail, copiez les modifications dans /chroot/sendmail/etc. De même /var/spool/mail devra pointer vers /chroot/sendmail/var/spool/mail de façon à ce que sendmail et les utilisateurs (quand ils se connectent) puissent voir les mêmes fichiers.

Le bon côté c’est que vous pouvez toujours envoyer du courrier, ce n’est que la réception qui pose un problème. Ainsi, j’ai pu installer sendmail avec apache sans difficulté. Certains de mes scripts Perl expédient du courrier et j’avais donc besoin que les fichiers sendmail soient copiés dans la zone "chrootée" d’apache.

D’autres choses à "chrooter".

Voici ma philosophie :

1. Tout devrait être "chrooté", y compris sendmail, ssh, apache, postgresql, syslog, et tous les services actifs sur l’ordinateur.
2. Tout devrait se trouver sous un compte non root (il est possible que vous soyez obligé de rediriger des ports protégés vers un port non protégé). C’est d’ailleurs valable pour sendmail et syslog.
3. Les logs devraient être envoyés à l’extérieur.
4. Une partition devrait être définie pour chaque service pour limiter l’espace disque qu’un pirate est susceptible d’utiliser s’il décide d’y écrire. Vous pourriez utiliser une interface loopback pour monter des fichiers en tant que système de fichier pour certains de ces services en cas d’insuffisance du nombre de partitions.
5. Root devrait être propriétaire des fichiers qui ne changent pas.

Pour ce qui concerne sendmail et syslogd, je persiste à croire qu’il pourrait être exécutés sous un compte non root. Pour sendmail, ce devrait être possible mais j’ai trouvé extrêmement difficile de l’exécuter sous un compte non root. Je n’y suis pas parvenu et je pense que c’est une grave erreur de ne pas pouvoir y arriver. Je sais que ça pose des problèmes, mais je crois qu’il faudrait penser à les résoudre. Puisqu’il est possible de gérer les permissions je ne vois pas pourquoi sendmail doit être exécuté en tant que root. Il y a sans doute des raisons que je ne perçois pas, mais je doute qu’on ne puisse pas surmonter les obstacles.

Pour syslog, je n’ai même pas essayé, mais je dirais que ces logs devraient être possédés par un compte non root et je ne vois pas pourquoi ce ne serait pas possible. Au moins, j’ai réussi à ce que syslog soit "chrooté" pour chaque service.

Tous les services devraient être installés sous des comptes non root. Même NFS. Tout.

Suggestions

- Utiliser deux logins pour ssh et prévoir deux démons sshd actifs.
- Trouver comment exécuter sendmail ou d’autres programmes de courrier sous des comptes non root.
- Supprimer les bibliothèques inutiles dans /lib. J’ai tout copié par paresse. Vous n’avez pas besoin de la plupart.
- Traiter les logs de syslogd en externe et voir si on peut lier syslogd à un port réseau et faire que tous les services se connectent à ce port réseau par l’interface loopback. Voir s’il est possible d’exécuter syslogd sous un compte non root.

Conclusion

Je pense que chroot est une bonne chose pour tous les services. Je crois que c’est une grave erreur de ne pas "chrooter" tous les services sous des comptes non root. J’aimerais que les distributions majeures le proposent, ou une distribution moins connue : n’importe laquelle. Mandrake a commencé en améliorant des choses de RedHat, peut-être que d’autres pourraient prendre des choses à Mandrake et améliorer chroot. Rien n’empêche quelqu’un de refaire le travail de quelqu’un d’autre dans le monde GNU/Linux, donc ce doit être réalisable. Si une société se décidait à tout "chrooter" et à créer un environnement simple permettant aux utilisateurs de gérer leurs services "chrootés", elle aurait une distribution fantastique ! Maintenant que Linux se répand, les gens ne veulent plus voir la ligne de commande, donc si tout est fait du côté de l’interface, ils n’ont plus à voir les entrailles et n’ont plus besoin de savoir ce qu’il se passe -- ils ont seulement besoin de configurer l’ensemble et de savoir que ça fonctionne !

Je suis à 100% pour l’idée que tous les services devraient être "chrootés" sous des comptes non root et qu’une distribution qui ne propose pas cette possibilité est impropre à l’utilisation dans un environnement de production. Je vais "chrooter" tout, autant que possible -- j’y arriverai peut-être.

J’envisage de créer un HOWTO sur "chroot". J’aimerais que quelqu’un m’aide à convertir cet article au format LyX pour pouvoir l’intégrer dans les HOWTO pour Linux.

Références

1. Si cet article change, il sera disponible sur http://www.gnujobs.com/Articles/23/chroot.html

---

Site Web maintenu par l´équipe d´édition LinuxFocus	Translation information:
© Mark Nielsen	en --> -- : Mark Nielsen (homepage)
"some rights reserved" see linuxfocus.org/license/	en --> fr: Georges Tarbouriech <georges.t/at/linuxfocus.org>
http://www.LinuxFocus.org	

2005-01-14, generated by lfparser_pdf version 2.51